Camel Transport for CXF

What's the Camel Transport for CXF

In CXF you offer or consume a webservice by defining its address. The first part of the address specifies the protocol to use. For example address=" http://localhost:9000" in an endpoint configuration means your service will be offered using the http protocol on port 9000 of localhost. When you integrate Camel Tranport into CXF you get a new transport "camel". So you can specify address="camel://direct:MyEndpointName" to bind the CXF service address to a camel direct endpoint.

Technically speaking Camel transport for CXF is a component which implements the CXF transport API with the Camel core library. This allows you to easily use Camel's routing engine and integration patterns support together with your CXF services.

Integrate Camel into CXF transport layer

To include the Camel Tranport into your CXF bus you use the CamelTransportFactory. You can do this in Java as well as in Spring.

Setting up the Camel Transport in Spring

You can use the following snippet in your applicationcontext if you want to configure anything special. If you only want to activate the camel transport you do not have to do anything in your application context. As soon as you include the camel-cxf-transport jar (or camel-cxf.jar if your camel version is less than 2.7.x) in your app, cxf will scan the jar and load a CamelTransportFactory for you.

xml<!-- you don't need to specify the CamelTransportFactory configuration as it is auto load by CXF bus --> <bean class="org.apache.camel.component. cxf.transport.CamelTransportFactory"> <property name="bus" ref="cxf" /> <property name="camelContext" ref="camelContext" /> <!-- checkException new added in Camel 2.1 and Camel 1.6.2 --> <!-- If checkException is true , CamelDestination will check the outMessage's exception and set it into camel exchange. You can also override this value in CamelDestination's configuration. The default value is false. This option should be set true when you want to leverage the camel's error handler to deal with fault message --> <property name="checkException" value="true" /> <property name="transportIds"> <list> <value>http://cxf.apache.org/transports/camel</value> </list> </property> </bean>

Integrating the Camel Transport in a programmatic way

Camel transport provides a setContext method that you could use to set the Camel context into the transport factory. If you want this factory take effect, you need to register the factory into the CXF bus. Here is a full example for you.

javaimport org.apache.cxf.Bus; import org.apache.cxf.BusFactory; import org.apache.cxf.transport.ConduitInitiatorManager; import org.apache.cxf. transport.DestinationFactoryManager; ... BusFactory bf = BusFactory.newInstance(); Bus bus = bf.createBus(); CamelTransportFactory camelTransportFactory = new CamelTransportFactory(); // set up the CamelContext which will be use by the CamelTransportFactory camelTransportFactory.setCamelContext(context) // if you are using CXF higher then 2.4.x the camelTransportFactory.setBus(bus); // if you are lower CXF, you need to register the ConduitInitiatorManager and DestinationFactoryManager like below // register the conduit initiator ConduitInitiatorManager cim = bus.getExtension(ConduitInitiatorManager.class); cim.registerConduitInitiator(CamelTransportFactory.TRANSPORT_ID, camelTransportFactory); // register the destination factory DestinationFactoryManager dfm = bus.getExtension(DestinationFactoryManager.class); dfm.registerDestinationFactory (CamelTransportFactory.TRANSPORT_ID, camelTransportFactory); // set or bus as the default bus for cxf BusFactory.setDefaultBus(bus);

Configure the destination and conduit with Spring

Namespace

The elements used to configure an Camel transport endpoint are defined in the namespace `http://cxf.apache.org/transports/camel`. It is commonly referred to using the prefix `camel`. In order to use the Camel transport configuration elements, you will need to add the lines shown below to the beans element of your endpoint's configuration file. In addition, you will need to add the configuration elements' namespace to the `xsi:schemaLocation` attribute.

Adding the Configuration Namespace<beans ... xmlns:camel="http://cxf.apache.org/transports/camel ... xsi:schemaLocation="... http://cxf.apache.org /transports/camel http://cxf.apache.org/transports/camel.xsd ...>

The `destination` element

You configure an Camel transport server endpoint using the `camel:destination` element and its children. The `camel:destination` element takes a single attribute, `name`, that specifies the WSDL port element that corresponds to the endpoint. The value for the `name` attribute takes the form *portQName*`.camel-destination`. The example below shows the `camel:destination` element that would be used to add configuration for an endpoint that was specified by the WSDL fragment `<port binding="widgetSOAPBinding" name="widgetSOAPPort">` if the endpoint's target namespace was `http://widgets.widgetvendor.net`.

camel:destination Element... <camel:destination name="{http://widgets/widgetvendor.net}widgetSOAPPort.http-destination> <camelContext id="context" xmlns="http://activemq.apache.org/camel/schema/spring"> <route> <from uri="direct:EndpointC" /> <to uri="direct:EndpointD" /> </route> < /camelContext> </camel:destination> <!-- new added feature since Camel 2.11.x <camel:destination name="{http://widgets/widgetvendor.net} widgetSOAPPort.camel-destination" camelContextId="context" /> ...

The `camel:destination` element for Spring has a number of child elements that specify configuration information. They are described below.

Element	Description
`camel-spring:camelContext`	You can specify the camel context in the camel destination
`camel:camelContextRef`	The camel context id which you want inject into the camel destination

The `conduit` element

You configure a Camel transport client using the `camel:conduit` element and its children. The `camel:conduit` element takes a single attribute, `name`, that specifies the WSDL port element that corresponds to the endpoint. The value for the `name` attribute takes the form *portQName* `.camel-conduit`. For example, the code below shows the `camel:conduit` element that would be used to add configuration for an endpoint that was specified by the WSDL fragment <`port binding="widgetSOAPBinding" name="widgetSOAPPort"`> if the endpoint's target namespace was `http://widgets.widgetvendor.net`.

xmlhttp-conf:conduit Element... <camelContext id="conduit_context" xmlns="http://activemq.apache.org/camel/schema/spring"> <route> <from uri="direct:EndpointA" /> <to uri="direct:EndpointB" /> </route> </camelContext> <camel:conduit name="{http://widgets/widgetvendor.net}widgetSOAPPort.camel-conduit"> <camel:camelContextRef>conduit_context</camel:camelContextRef> </camel:conduit> <!-- new added feature since Camel 2.11.x <camel:conduit name="{http://widgets/widgetvendor.net}widgetSOAPPort.camel-conduit" camelContextId="conduit_context" /> <camel:conduit name="*.camel-conduit"> <!-- you can also using the wild card to specify the camel-conduit that you want to configure --> ... </camel:conduit> ...

The `camel:conduit` element has a number of child elements that specify configuration information. They are described below.

Element	Description
`camel-spring:camelContext`	You can specify the camel context in the camel conduit
`camel:camelContextRef`	The camel context id which you want inject into the camel conduit

Configure the destination and conduit with Blueprint

From **Camel 2.11.x**, Camel Transport supports to be configured with Blueprint.

If you are using blueprint, you should use the the namespace `http://cxf.apache.org/transports/camel/blueprint` and import the schema like the blow.

Adding the Configuration Namespace for blueprint<beans ... xmlns:camel="http://cxf.apache.org/transports/camel/blueprint" ... xsi:schemaLocation="... http://cxf.apache.org/transports/camel/blueprint http://cxf.apache.org/schmemas/blueprint/camel.xsd ...>

In blueprint `camel:conduit` `camel:destination` only has one camelContextId attribute, they doesn't support to specify the camel context in the camel destination.

<camel:conduit id="*.camel-conduit" camelContextId="camel1" /> <camel:destination id="*.camel-destination" camelContextId="camel1" />

Example Using Camel as a load balancer for CXF

This example shows how to use the camel load balancing feature in CXF. You need to load the configuration file in CXF and publish the endpoints on the address "camel://direct:EndpointA" and "camel://direct:EndpointB"

{snippet:id=example|lang=xml|url=camel/trunk/examples/camel-example-cxf/src/main/resources/org/apache/camel/example/camel/transport/CamelDestination.xml}

Complete Howto and Example for attaching Camel to CXF

Better JMS Transport for CXF Webservice using Apache Camel